AD-A101 708 NAVAL AIR ENGINEERING CENTER LAKEHURST NJ F/G 9/2
INTRODUCTION TO CURRENT COMPUTER AIDS TO DIGITAL TEST DESIGN FO--ETC(U)
FEB 74 F LIGUORI

UNCLASSIFIED NL

1 OF 1
AD A
101708

END
DATE
FILMED
8-81
DTIC

LEVEL II

Published at Project SETE Workshop, Orlando, Fl. Feb 1974

DTIC
ELECTE
JUL 21 1981
S D

AD A101708

INTRODUCTION TO CURRENT COMPUTER AIDS TO DIGITAL TEST DESIGN FOR AUTOMATED TEST EQUIPMENT

Fred Liguori, ATE Branch Head
Naval Air Engineering Center, Lakehurst, NJ 08733

PROBLEM DISCUSSION

The concept of utilizing a computer as an aid for test program design and validation is not new to automatic testing technology. During the 1960's a number of attempts were made to develop simulators to simplify the test program debugging and validation effort. Most, if not all, such efforts proved to be less than effective. The complexity of the units to be tested generally made modeling prohibitively expensive. Attempts at modeling circuit designs to generate test programs with computers were frustrated by the same difficulties plaguing automated circuit design systems; the models required extensive libraries of component characteristics which were extremely costly to develop and very difficult to express mathematically.

Just about when most ATE experts became convinced that simulation was not a practical tool for their technology, the complexion of the unit under test began to change rapidly. Digital techniques became a major factor in electronics equipment design. Micro-miniaturization became a reality. The hand-crafted, unstructured art of analog circuit design began giving way to standardized pulse-oriented circuits. Now, however, it took a myriad of digital devices in a maze of logic circuitry to replace very few analog components which did the same job, but with less precision. Test designers not only had to re-orient themselves to think digitally, but also found that a typical module no longer contained about a hundred discrete components but rather almost a hundred digital networks. Whereas each analog component might require two or three tests, each digital network could require fifty tests or more just to assure correct performance. With this magnitude of testing, it was no longer sufficient to have a nice software system to translate test designs into ATE machine language. A test designer simply was unable to generate the tests in a realistic time frame at a tolerable price.

Fortunately, digital circuit complexity is off-set by digital element simplicity and regularity. The problem of test design has therefore transitioned from coping with circuits containing relatively few hard to model analog devices to extremely large quantities of easy to model digital devices. Computerized simulation thus becomes not only practical, but in many cases, the only way to generate adequate tests for the more complex digital unit under test. Computer aided test design tools have therefore become a requirement rather than a nicety. Understanding these new software tools is essential for today's ATE test designers and systems engineers. This paper introduces some of the more successful, current systems for computer aided test design. The discussion is limited to digital test design because computer aid is

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

mandatory as well as most practical for digital applications.

DIGITAL TESTING TECHNIQUES

Before discussing automated techniques for digital test design, it might be well to review testing techniques and terminilogy. Digital testing can be broadly divided into static and dynamic tests. Dynamic tests are those performed under conditions equivalent to normal operation of the circuit. It implies stimulating the circuit under test with appropriate power supplies and input signals of the same pulse repitition rate and other characteristics found during normal operation. Outputs are also tested for normal pulse characteristics. Static testing involves all tests other than dynamic as defined. It should be noted that even when voltages and signals are applied to the circuit, tests are still classified as static if the pulse rate falls outside the normal operating range.

Generally static testing is performed at substantially slower pulse rates than normal operating rates. Static tests can be subdivided into three types:

1. Continuity Testing. This is simply checking for open or short circuits. Other resistance measurements, performed while the circuit is not energized, are also included in this category.

2. DC Testing. This category of tests applies voltages or current sources to the circuit or unit under test (UUT) two pins at a time. This tests the ability of the selected pin pair to transmit power under proper load conditions. All input and output signals are thus sequentially tested.

3. Functional Testing. These tests check the transfer function of the circuits at a rate generally much slower than the normal pulse rate. They attempt to verify the ability of a circuit to combine and/or separate logic states and pulses according to a defined "truth table." The truth table defines the input and output logic words (represented by groups of ONEs and ZEROs) required for normal circuit performance. Since these tests verify the logical operation of the circuit, they are also classified as logic tests. Static logic tests check truth table outputs after the circuit has reached steady-state conditions. This is also referred to as functional testing.

Dynamic tests can be subdivided into two types:

(1) Logic Tests

(2) Parametric Tests

Dynamic logic tests are based on truth table values just as with static logic tests. The difference is that dynamic tests are performed at the normal pulse rates of the circuit rather than waiting until the circuit reaches steady-state logic conditions. In other words, tests are made with precise consideration of time relationships under clocked conditions. In more exhaustive testing schemes, the clock or pulse rate and/or amplitude are varied between defined limits to assure performance within specified variations during operating conditions. This is called marginal testing.

Parametric tests are measurements relating to the shape of the pulses, such as rise time, fall time, overshoot on pulse duration. To be meaningful they must be performed under dynamic conditions. They are, in effect, analog measurements of digital signals. Parametric tests are rarely performed for maintenance testing because proper performance of dynamic logic tests depend on intollerance parameters. Hence, proper performance of dynamic tests implies proper performance of significant parameters.

TESTING REQUIREMENTS

Testing requirements are determined by both the nature of the circuit being tested and the purpose of the test. There are many purposes for tests but these can be grouped into three categories:

(1) Design testing

(2) Production testing

(3) Maintenance testing

Design requires the most comprehensive testing because it involves the first formulation of a circuit. It requires that all characteristics be considered, dynamic as well as static. Production testing assumes the existance of a proven design but cannot assume that all components are operating or that they have been correctly assembled. Maintenance testing can assume that the circuit design is correct and that at least once it was in good working order. Obviously it is important to keep the purpose of the test in mind when deciding how to test a circuit. Although much of the testing theory is valid for all categories of testing, failure modes vary so the approach to testing can be substantially different for each category. This paper is primarily concerned with maintenance testing so unless otherwise indicated, assumptions and conclusions stated are based on maintenance testing. Primary concerns are therefore limited to the nature of the circuit (how is it designed and what does it do) and its failure modes (how does it fail and how are these failures manifested).

The nature of digital circuits today is that most are designed to discriminate between and operate on combinations of digital signals. They are most sensitive to the presence or absence of signals but generally not too concerned with the shape or timing of the signals. Hence they are called combinational logic circuits. It is therefore sufficient to test combinational circuits in current usage are of functional) tests. It seems that in excess of 90% of all digital circuits in correct usage are of the combinational variety.

Whether static logic tests truly suffice for testing combinational logic depends on the failure modes of the components from which the circuits are built. Most components are either discrete semiconductors or elements combined in a integrated circuit (IC) chip. Chips also consist of semiconductive materials so they tend to exhibit failure modes similar to discrete semiconductors. The biggest difference is that chips are more prone to multiple gate failures because of the close proximity of their elements. In general, though, semiconductors used in digital circuits behave as gates which can be either open or closed but not part way in between. Hence the usual failure modes are "stuck at one" or "stuck at zero." In positive logic the one is the presence of the signal and zero is the absence of the signal. Because there are generally wide latitudes in signal level and duration within which the circuit will operate, exact timing or pulse shapes are inconsequential. There is, however, a minimum value or threshold outside of which a signal is no longer considered a one. If the component involved tend to degrade rather than fail catestrophically (stuck at one or zero) then even combinational logic could require dynamic or parametric testing to assure proper performance. Generally, however, the nature of digital components and circuit designs are such as to avoid degradation effects. To the degree that designers are successful in this endeavor, then static testing of combinational logic circuits is adequate. This should be determined during design testing, not during maintenance testing.

Although they are the exception, some circuits are necessarily sensitive to the shape or timing of the digital signals. These are referred to as time dependent or sequential logic circuits. Dynamic testing is required for sequential logic circuits. Whether parametric tests are also required per se, or whether the dynamic tests imply within tolerance parameters is somewhat controversial. It seems clear, however, that if dynamic tests fail, it generally requires parametric tests to determine the cause of failure even though they may not be required simply to conclude that a failure exists. Hence, the real determinant as to whether parametric testing is required for sequential logic testing is how much one wishes to know about the nature of the failure.

Even in the relatively rare circuits where dynamic and/or parametric testing is required, only a small percentage of the tests must be made dynamically. The large preponderance of tests are concerned with combinational logic that requires only static logic testing. The problem in digital testing today is primarily one of bulk rather than individual sophistication.

Digital testing inherently requires a large number of test patterns to test all circuit functions. For example: a 200 IC (Integrated Circuit) Shop Replaceable Assembly (SRA) will typically require 1,000 to 4,000 test patterns to detect 95 percent of the 5,600 failures that could develop in an SRA of this size. In addition, it would require 40,000 36-bit words of fault dictionary to relate circuit response patterns to specific faults. With a computer generated test program, a 200 IC SRA could be tested on commercial automatic test equipment in less than a minute with a 95 percent fault detection and an average fault isolation of 1.5 IC's. This is assuming the test patterns and test program are automatically generated by computer. On the other hand, manually generated test patterns and test programs for a 200 IC SRA would produce a 20 to 30 percent fault detection and a fault isolation of 6 to 10 IC's, and take longer than a minute of test time on ATE. The real problem in the manual test pattern/test program approach is not knowing the true percent fault detection. In manual test program preparation it is almost impossible to determine the actual level of percent fault detection when the SRA exceeds 50 IC's in size. The only economical way to prove or disprove a claimed percent detection level in large SRA's is through computer simulation techniques, the same techniques that should be used to generate the test programs in the first place.

Fortunately, the combinational logic which represents the vast majority of digital circuitry of today and the near future, is relatively easy to model. Unlike analog circuits which require complex mathematical formula derived through extensive emperical analyses, digital logic elements are quite simple. Furthermore, there is a rather small number of logic circuit types involved in most digital systems. The complexity lies in the combination of logic elements not in the basic design or characteristics of the elements. Thus the logical approach to digital simulation is to build up a library of basic element characteristics in the simulation model. These elements are then copied and linked into the model of the system to be simulated. Such simulation techniques form the basis of all effective computer aided circuit design and testing systems. In fact, some computer aided test design systems are adaptations of circuit design systems. This paper is concerned only with simulation for test design.

COMPUTER AIDED TEST DESIGN SYSTEMS

Many systems are available today for the use of computers in digital test design. Table 1 identifies eleven systems in current usage which have enjoyed successful applications in industry. Although

Table 1 Some Computer Aided Test Design Systems for Digital Circuitry

CAPP System	Company Used By	All Failure Modes	Fault Detection	Fault Isolation Dictionary	Automatic Stimulus Generation	**Three** State Simulation	Initial-ization Capability	Asynchro-nous Capability	Component Library	Race and Hazard Capability	True Signal Timing Capability	Signal Spike Evaluation Capability	Fail-All and Fail Detect Mode	Simple Modeling Techniques (method)
LASAR II	LTV	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes (Nand equiv.)
D-LASAR	Digitest Lockheed, Ga. Grumman	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes (Nand equiv.)
FAIR SIM II FAIRGEN.	Fairchild	Yes	Yes	Yes	Limited	Yes	Yes	Limited	Yes	Limited	Yes	Yes	Yes	Yes (macro)
TESTAID	Telpar Hewlett-Packard	No	Yes	Yes	Limited	Yes	Limited	Limited	Yes	Limited	No	No	Yes	Yes (macro)
TESGEN COMTEST	Westinghouse AAI	No	Yes	Yes	Limited	Yes	Limited	No	Yes	Limited	Yes	No	Yes	Yes (Nand equiv.)
		No	Yes	Yes	Limited	Yes	Yes	Yes	Yes	NA	NA	NA	NA	Yes (macro)
FLASH	Mirco Inc. Sperry-Rand General Radio	No	Yes	Yes	No	No (2 state)	No	No	Yes	No	Yes	No	Yes	Yes (macro)
TGEN	RCA	No	Yes	Yes	No	Yes	Limited	Yes	Yes	Limited	Yes	No	Yes	Yes (Nand macro)
FAULTS II	General Dynamics	No	Yes	Yes	Yes	Yes	Limited	Yes	Yes	Limited	Yes	No	Yes	Yes (macro)
SALT	IBM	No	Yes	Yes	Yes	Yes	Limited	Limited	Yes	Limited	Yes	No	Yes	Yes (macro)
TASC	Pacific Applied Systems, Inc.	No	Yes	Yes	No	Yes	No	No	Yes	Limited	Yes	No	No	Yes (macro)
SATGEN	Hughes Aircraft Co.	No	Yes	Yes	No	Yes	No	No	Yes	No	No	No	No	Yes (macro)

NA Information Not Available

this is not a comprehensive listing, it is believed that this list embodies the most used and most successful systems available today. Table 1 lists key features and characteristics of each system for summary comparison purposes. It is not intended to provide a value comparison for the most effective system depends entirely on the application involved. Rather than attempting to rate these systems, this paper points out some of the major attributes of each system. It is up to the potential user to judge the relative advantages for his application.

LASAR

This system derives its name from Logic Automated Stimulus and Response. It was originally developed by LTV Corporation of Ft. Worth Texas under Navy contract. This version, called LASAR II was made available to contractors preparing test programs to be executable on the VAST system. However, the test design system was designed to be tester-independent. Most users have converted from LASAR II to more recent versions to take advantage of improved programming ease, reduced computer operation costs and more optimum code generation. However, the basic approach for D-LASAR is fundamentally the same as for LASAR II. Differences in organization and capabilities of the two LASAR versions are shown diagramatically in Figures 1 and 2.

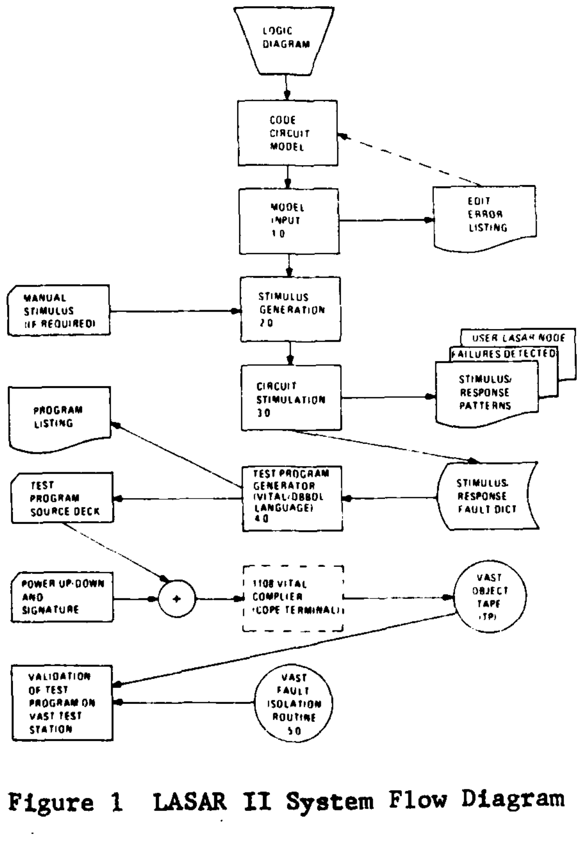
LOGIC DIAGRAM
CODE CIRCUIT MODEL
MODEL INPUT 1.0
EDIT ERROR LISTING
MANUAL STIMULUS (IF REQUIRED)
STIMULUS GENERATION 2.0
USER LASAR NODE
FAILURES DETECTED
STIMULUS/RESPONSE PATTERNS
CIRCUIT SIMULATION 3.0
PROGRAM LISTING
TEST PROGRAM SOURCE DECK
TEST PROGRAM GENERATOR (VITAL/DBBOL LANGUAGE) 4.0
STIMULUS RESPONSE FAULT DICT
POWER UP-DOWN AND SIGNATURE
1108 VITAL COMPILER (COPE TERMINAL)
VAST OBJECT TAPE (TP)
VALIDATION OF TEST PROGRAM ON VAST TEST STATION
VAST FAULT ISOLATION ROUTINE 5.0

Figure 1 LASAR II System Flow Diagram

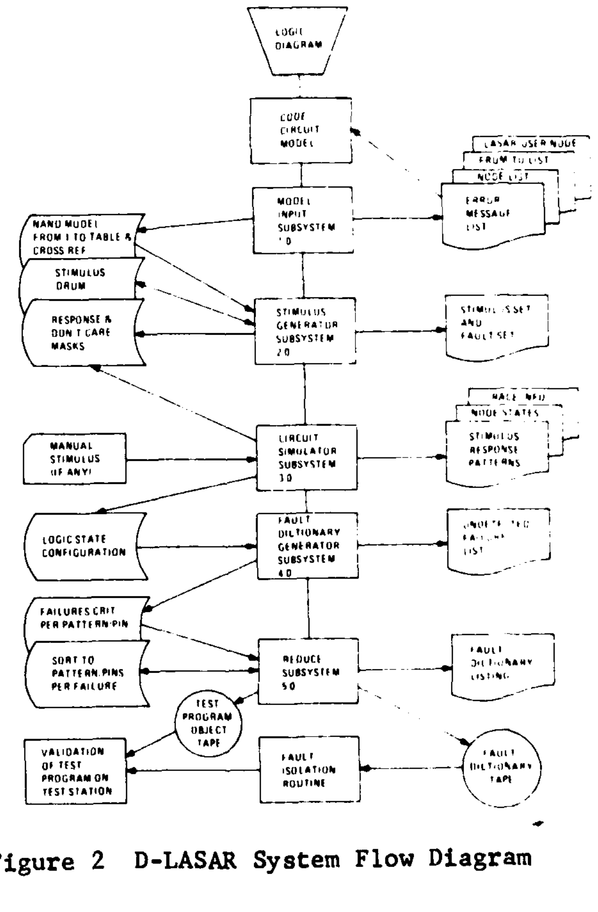
LOGIC DIAGRAM
CODE CIRCUIT MODEL
LASAR USER NODE
FROM TO LIST
NODE LIST
ERROR MESSAGE LIST
NAND MODEL FROM 1 TO TABLE & CROSS REF
MODEL INPUT SUBSYSTEM 1.0
STIMULUS DRUM
RESPONSE & DON'T CARE MASKS
STIMULUS GENERATOR SUBSYSTEM 2.0
STIMULUS SET AND FAULT SET
MANUAL STIMULUS (IF ANY)
CIRCUIT SIMULATOR SUBSYSTEM 3.0
NODE STATES
STIMULUS RESPONSE PATTERNS
LOGIC STATE CONFIGURATION
FAULT DICTIONARY GENERATOR SUBSYSTEM 4.0
FAILURES CRIT PER PATTERN PIN
SORT TO PATTERN PINS PER FAILURE
REDUCE SUBSYSTEM 5.0
FAULT DICTIONARY LISTING
TEST PROGRAM OBJECT TAPE
VALIDATION OF TEST PROGRAM ON TEST STATION
FAULT ISOLATION ROUTINE
FAULT DICTIONARY TAPE

Figure 2 D-LASAR System Flow Diagram

LASAR II has five major subsystems:

- Model Input (1.0)
- Stimulus Generation (2.0)
- Circuit Simulation (3.0)
- Test Program Generation (4.0)
- Fault Isolation (5.0)

The model input subsystem performs the following functions:

a. Interpret/edit the user model cards.

b. Add new NAND equivalents to the component library.

c. Arrange the order of components for minimum computer run-time.

d. Expand components into NAND equivalents.

e. Build user node to LASAR node cross-reference table.

f. Build node pointer and FROM/TO table.

The stimulus generator subsystem provides:

a. Generation of stimulus sufficient to detect greater than 95 percent of all possible failures (including component internal failures).

b. Generate a one-pin response per stimulus set.

c. Derive critical NAND node-component on-off's (i.e., failure detected) per stimulus set.

The circuit simulation subsystem performs the following functions:

a. Derives the output response patterns for all forced output nodes.

b. Derives the failures which each stimulus/response pair will detect.

The fault isolation subsystem is designed to be used with the Circuit Simulation Subsystem or automatic test equipment to provide total fault detection/isolation capability. Two fault classifications are defined: "stuck" inputs and "stuck" outputs. "Stuck" refers to a fault which causes a signal to be permanently at a logic 1 or 0 commonly referred to as SA1 or SA0. For example, an open input diode on a DTL gate will appear as though that input were stuck at 1 (SA1). Outputs, defined as signals available to an external device such as a tester, are examined after every input change. When an output of a faulted network is different from the good network, and this difference does not involve a signal in the unknown state, the fault is assumed to have been detected. Defects may not be detected for two reasons - redundant logic (circuit design) or incomplete test sequences (inadequate stimulus patterns).

The Test Program Generator subsystem is based on a general test technique that provides static or functional tests for any digital UUT and consists of applying stimulus patterns; recording response patterns; comparing actual response patterns with "expected" response patterns to detect errors; and identifying and storing detected error patterns. The output of the test program generator is a program listing and the test program source card deck in the language of the automatic test equipment source programming language (i.e., VITAL, BASIC, ATLAS, etc.). The user completes the test program by adding the required UUT signature test and power application instructions to the source program card deck. The combined source deck is then input on the cope terminal and compiled using the Vital compiler to produce a VAST object magnetic tape of the test program.

The major advantages of D-LASAR over LASAR II are due primarily to the improvements in the design of the programs for stimulus generation, circuit simulator, fault dictionary generator, and the addition of the reduce subsystem to greatly decrease unnecessary response data. The net result of the D-LASAR improvements over LASAR II is approximately 100:1 reduction in computer run-time. For example, assuming a 2μsec computer cycle time (1108) and an experienced user, the following computer run-times are typical for a 200 IC network:

- ● Input Subsystem: 7 runs = 7 minutes
- ● Stimulus Generator: 30 to 120 minutes total
- ● Simulator: 3000 patterns twice = 2 minutes
- ● Fault Dictionary and Reduce: 300 patterns = 50 minutes

Total run-time would be in the range of 1.5 to 3 hours and includes multiple stimulus generator segment passes to detect and clean up any work-arounds.

FAIRSIM II/FAIRGEN

FAIRSIM II provides a computer simulation of digital circuits, with primary orientation toward computer aided design (CAD) of MSI and LSI designs. FAIRSIM II is an improved version of the Fairchild Digital Simulation program; its relationship to the complete LSI CAD program is shown in Figure 3. A proposed LSI design is coded into FAIRSIM's language or MACRO description along with a test sequence. After simulation, the FAIRSIM data becomes the basis for computer aided cell placement, interconnection, and acceptance test generation programs. FAIRSIM uses the "Look Ahead" simulation technique. A digital system consists of interconnected logic elements such as gates, flip-flops, etc. The interconnections transfer binary signals between logic elements. In a typical digital

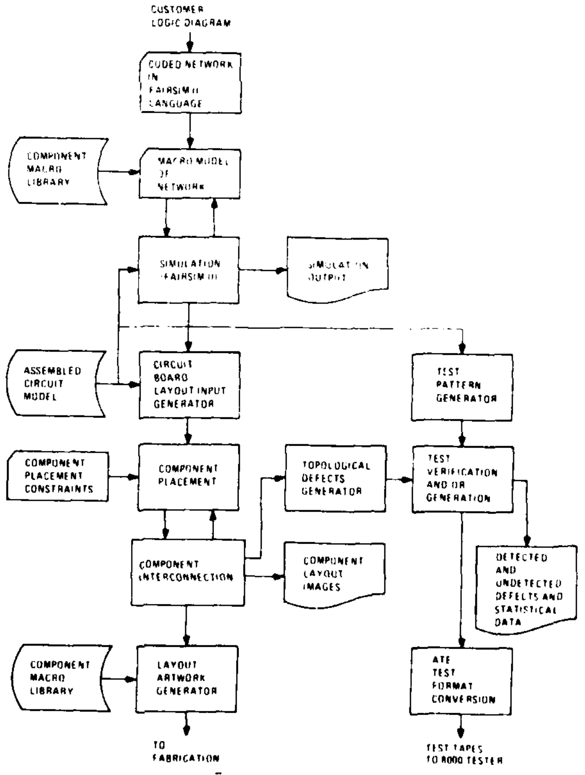
CUSTOMER LOGIC DIAGRAM
CODED NETWORK IN FAIRSIM II LANGUAGE
COMPONENT MACRO LIBRARY
MACRO MODEL OF NETWORK
SIMULATION (FAIRSIM II)
SIMULATION OUTPUT
ASSEMBLED CIRCUIT MODEL
CIRCUIT BOARD LAYOUT INPUT GENERATOR
TEST PATTERN GENERATOR
COMPONENT PLACEMENT CONSTRAINTS
COMPONENT PLACEMENT
TOPOLOGICAL DEFECTS GENERATOR
TEST VERIFICATION AND OR GENERATION
COMPONENT INTERCONNECTION
COMPONENT LAYOUT IMAGES
DETECTED AND UNDETECTED DEFECTS AND STATISTICAL DATA
COMPONENT MACRO LIBRARY
LAYOUT ARTWORK GENERATOR
ATE TEST FORMAT CONVERSION
TO FABRICATION
TEST TAPES TO 8000 TESTER

Figure 3 Fairchild System Flow Diagram

system operation, only a small percentage of the elements change state during any short period of time. In the "Look Ahead" technique advantage is taken of this slow state of change to provide simulation of a circuit element. Logic transitions from one state to the next are assumed to be instantaneous. Thus, input threshold or noise immunity cannot be simulated in FAIRSIM directly.

The FAIRSIM II simulator is a 3 state simulator. Elements may have outputs which are 1, 0 or undefined. While the simulator is in operation the programs perform the following:

a. It transmits the new output along the circuit fan-out net as inputs to other logic elements within the model.

b. It then stimulates all affected elements in the model to determine if their outputs will change state, and

c. If they will change state, it places the proper flag in the time sequence unit at the time slot their output is due to change (i.e., after their delay)

d. The scanning of the time sequence continues and also updates total elapsed system time before recycling.

The output of FAIRSIM is a list of requested logic element outputs, either 1, 0, or undefined. This listing corresponds to the equivalent of a multitrace oscilloscope.

The Fairchild Test Generation (FAIRGEN) verification program is used in the production of functional test programs for digital SRA's, with primary orientation toward micromosaic arrays. The test verification function of the program is used to determine the effectiveness of a given test sequence. The test generation capability is used to generate additional test patterns to increase the percentage of fault detection to more thoroughly exercise and test the SRA in question. The purpose of the test generation process, then, is to produce a series of input patterns, or tests, which when applied to the SRA under test, yield differing outputs for good and faulted SRA's. The type of defects or faults considered by FAIRGEN on a particular run are selected by the user. The four possible defect types are:

- Stuck Outputs - Each defect of this type consists of the output of one logic block (gate) being "stuck" at a particular value (either SA1 or SA0).

- Stuck Inputs - Each defect consists of one input to one gate stuck at one or zero (i.e., the gate behaves as if that input were connected to a source of a constant 1 or 0).

- Shorts - Each defect consists of the output of one block shorted to the output of another (the short is considered to behave logically as a "wired and" or "wired or" function).

- Opens - Each defect of this type consists of a "break" or "open" in the wire(s) connecting a particular block to some combination of its driven blocks. The inputs to the driven blocks which are left unconnected are considered to be stuck at 1 or stuck at 0. The FAIRGEN Program has been designed to be compatible with the FAIRSIM program and provisions have been made for calling FAIRGEN directly from FAIRSIM. In this case, the test sequence defined by the simulation is stored on disk and at the conclusion of simulation, FAIRGEN is called. A well-defined programming interface is provided to convert test sequences produced by FAIRGEN into formats suitable for input to automatic test equipments.

The total system concept for digital module testing on automatic test equipment is presented in Figure 4.

TESTAID

The TESTAID system is a product of TELPAR INC. Its flow is presented in Figure 5. The system is designed to produce test programs for digital SRA's. The SIGLIST program edits the SRA logic description or model for errors. This program accepts a description of the logic (at the IC or package level), adds the device description MACRO from the library, and performs extensive error checking.

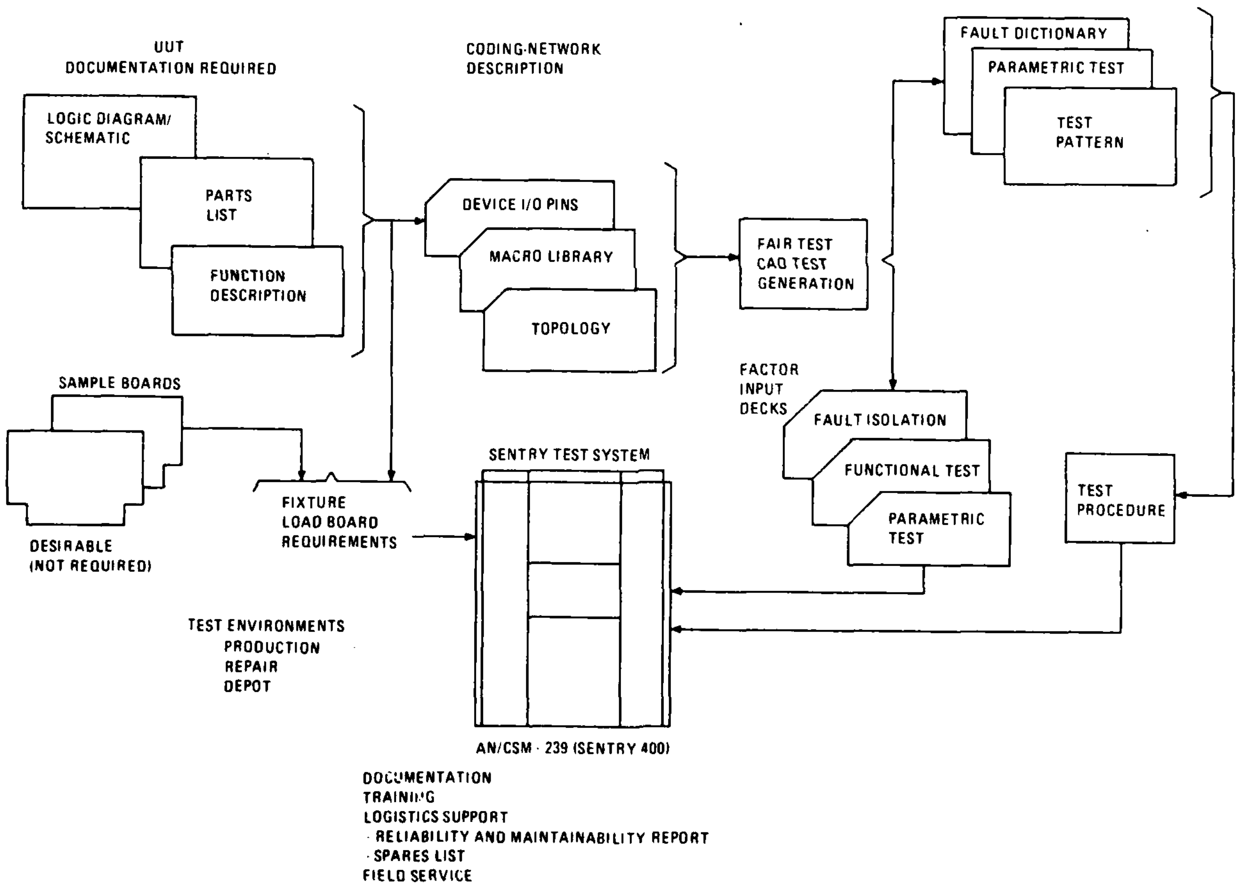

Figure 4 Fairchild Total System Concept for Digital Module Test Generation

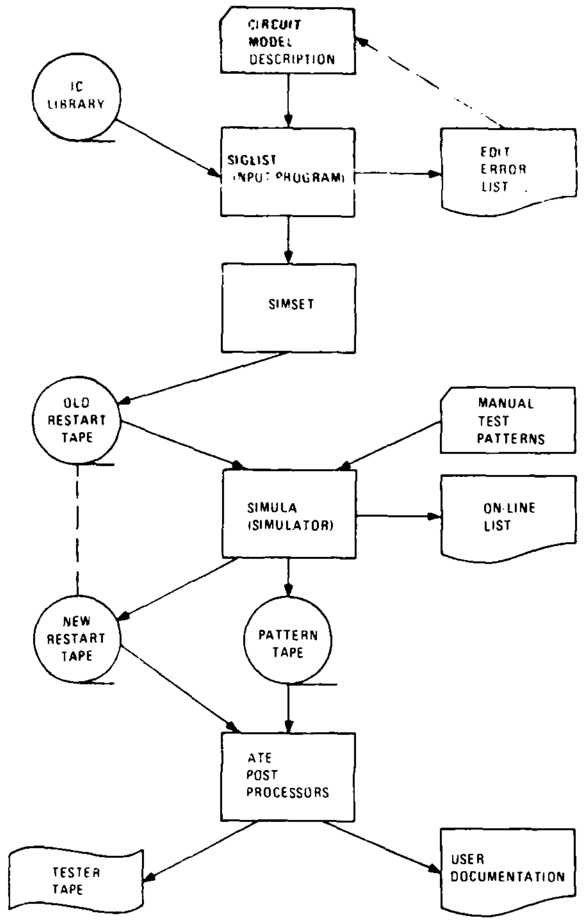

Figure 5 TESTAID Flow Diagram

The principal output of SIGLIST is an encoded form of the input that is organized for use by later programs. The machine readable form of the SRA logic produced by SIGLIST is input to a program called SIMSET. SIMSET produces as its output a reorganized form of the SRA logic in the format needed by the simulator. In addition, it produces a table of faults. This fault table is minimized to include only one fault from each set of equivalent faults (from the standpoint of the given output pins). The set of faults normally produced consist of the stuck-at-0 (SA0) and stuck-at-1 (SA1) faults on:

- Board output pins
- Board input pins
- Package output pins
- Package input pins

Output from SIMSET is immediately ready for the simulator, called SIMULA. In addition to the logic input from SIMSET, which is called a restart tape, manual test pattern sequences are input to SIMULA. The simulator then simulates all of the faulty circuits defined by the faults present on the input restart tape against all of the input patterns. When a fault is detected, it is removed from the table of faults, and simulation of that fault is stopped. After simulation is completed, a new restart tape is produced, with the same format as that produced by SIMSET except for two differences. There are usually fewer faults in the fault table, and there are now internal states for each of the faults in the table, so that the exact internal configuration of each circuit fault is preserved.

Another output of SIMULA is a tape with the input patterns, output patterns for the good responses, all of the internal signals for good responses, and the output patterns for each of the faults that differ from the good circuit at this pattern step. This tape is used by the post-processor to produce a test program compatible with the user's ATE.

TESTAID is programmed in PL360 to be run on the IBM 360 computer. The system has run on almost all models of the IBM 360 computer from a model 25 (48K) to a model 195 ($4K^2$). The operating system plus the entire SIMULA code (no overlays are used) amounts to slightly over 27K bytes, leaving 37K bytes for data storage in a 64K machine. In practice, the most commonly used machine is a model 30 with 64K bytes (16K words of 32 bits). A machine of this size can handle a digital circuit of 1200 gates with reasonable computer run-time.

COMTEST/TESTGEN

COMTEST is a group of computer programs developed by Westinghouse Electric Corp. to generate digital test patterns and fault analysis. Digital circuits are described using macros. The simulator is a three valued (1, 0, and (don't know)) as an option, or two valued (1 and 0). Stimulus patterns are generated manually or automatically and are inputted to the simulator which produces the good responses that are compared to faulting conditions in combination or separately. Fault analyses includes:

- Outputs stuck at 1
- Outputs stuck at 0
- Inputs stuck at 1
- Inputs stuck at 0
- Externals stuck at 1
- Externals stuck at 0
- Fault internal macros
- Signals stuck at 1 and 0

With single pass runs the simulator handles circuits with up to 250 input pins, 1000 output pins, and 1000 circuit elements. Larger circuits are run in multiple passes. This system is available via Westinghouse's Time-Sharing system on a UNIVAC 1108. TESTGEN is an automatic test program generation system (Figure 6) utilizing a three valued unit-delay digital fault simulator and automatic stimulus generator to produce test programs for testing SRA's containing as many as 2,000 circuit elements. The simulator explodes all circuit components to their NAND equivalents and maintains a library of components. The stimulus generator automatically generates initialization sequences to bring the model to a know state. Stimulus patterns are generated automatically utilizing a reverse track algorithm to find test patterns. The algorithm is very efficient for combinational logic but fails when applied to asynchronous sequential and memory type circuits.

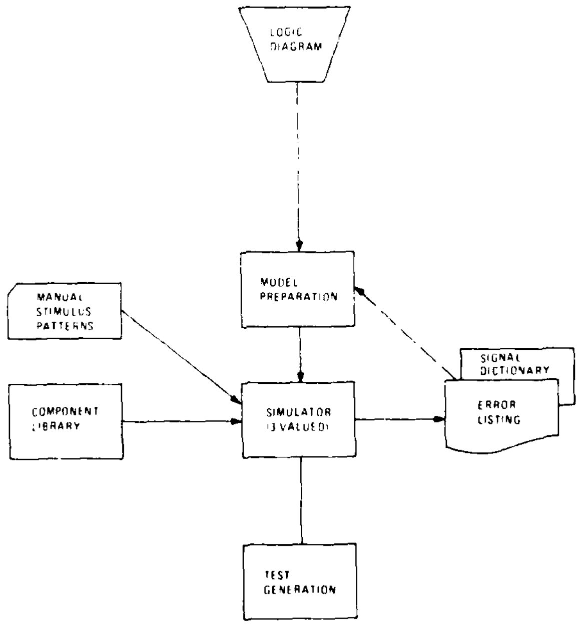

Figure 6 TESTGEN System Flow Diagram

The procedure in producing test programs involves: model preparation, simulation, and test generation. Model preparation consists of labeling the circuit schematic of a given SRA. This consists of assigning signal names to the input and output pins and to internal wires on the circuit schematic. Next comes coding the schematic and stimulus information in a language understood by the simulator. Concurrently, a signal dictionary list is coded for the automatic test program generator. This dictionary contains all connectors as well as the IC-pin designations associated with each signal name. The initial run through the simulator edits the accuracy of the model description by comparing the simulated responses to the true responses for all stimuli used in the test. After coding errors have been eliminated the simulation is continued and diagnostic data is generated. When the diagnostic information is sufficiently high (90 percent detection or better) it is passed on for test program generation. However, when the detection is low, the test engineer must determine if manually generated stimuli would reduce the list of undetected faults.

If it can be reduced by manual stimuli, simulation is continued until the desired detection and isolation resolution is achieved. The main body of the test program is conceived from data produced by the simulator which considers the effects of faults on the logic behavior of each digital circuit independent of the class of switching circuitry being tested (DTL, TTL, FET, etc.). These logical faults are represented by an input or an output of some gate being stuck-at-one (SA1), input/output shorted to high voltage, or stuck-at-zero (SA0), input/output shorted to a low voltage level.

The AAI Corporation uses Westinghouse's TESTGEN to generate test data as inputs to their test program generator DETOLGEN for compatible outputs to the 5500 ATE. The AAI CAD system is presented in Figure 7.

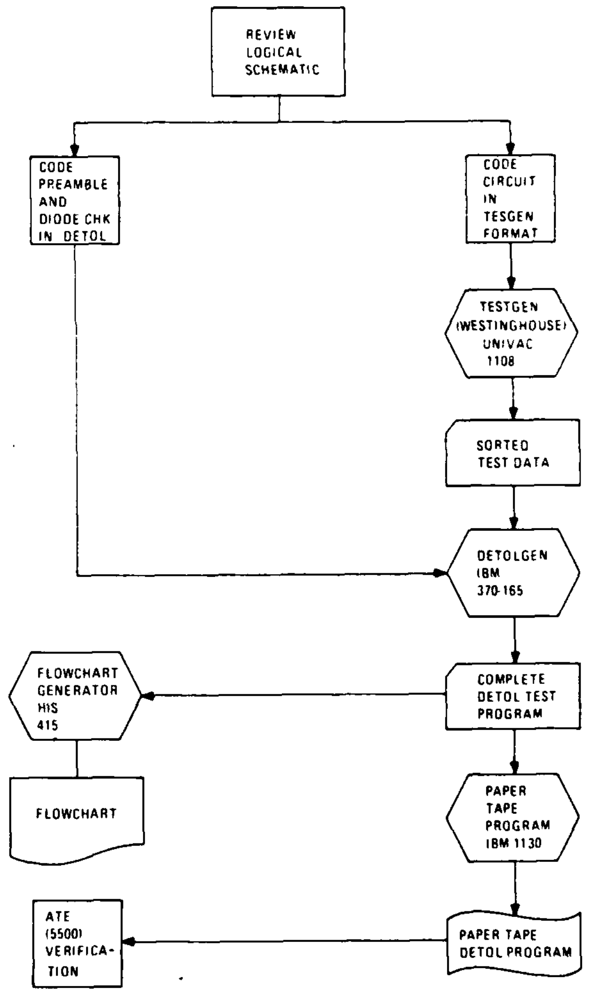
REVIEW LOGICAL SCHEMATIC
CODE PREAMBLE AND DIODE CHK IN DETOL
CODE CIRCUIT IN TESGEN FORMAT
TESTGEN (WESTINGHOUSE) UNIVAC 1108
SORTED TEST DATA
DETOLGEN IBM 370-165
COMPLETE DETOL TEST PROGRAM
FLOWCHART GENERATOR HIS 415
FLOWCHART
PAPER TAPE PROGRAM IBM 1130
PAPER TAPE DETOL PROGRAM
ATE (5500) VERIFICA-TION

Figure 7 CAD System Flow Diagram

FLASH

The FLASH (Fault Logic and Simulation Hybrid) system (Figure 8) was developed by Micro Inc. It is

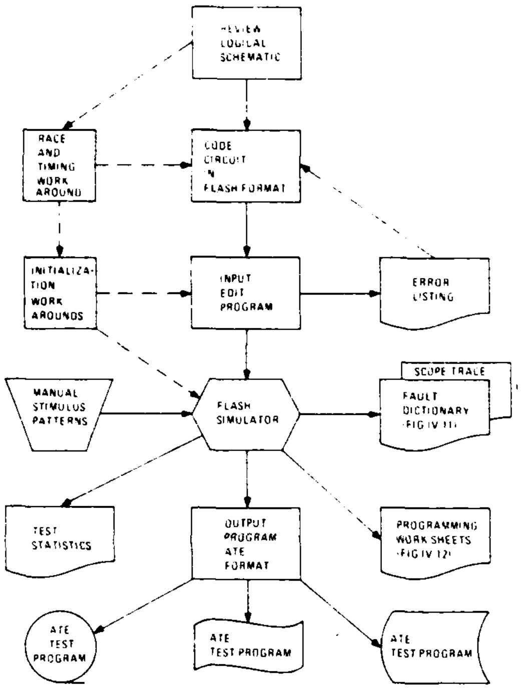
REVIEW LOGICAL SCHEMATIC
RACE AND TIMING WORK AROUND
CODE CIRCUIT IN FLASH FORMAT
INITIALIZATION WORK AROUNDS
INPUT EDIT PROGRAM
ERROR LISTING
MANUAL STIMULUS PATTERNS
FLASH SIMULATOR
SCOPE TRACE
FAULT DICTIONARY (FIG IV 11)
TEST STATISTICS
OUTPUT PROGRAM ATE FORMAT
PROGRAMMING WORK SHEETS (FIG IV 12)
ATE TEST PROGRAM
ATE TEST PROGRAM
ATE TEST PROGRAM

Figure 8 Flash System Flow Diagram

a two-state (1 and 0) simulator designed to produce good and faulted responses for simulated digital circuit boards. The system is programmed in Fortran IV and is designed for the maintenance and manufacturing environment. This system provides a computer aided maintenance tool to assist the maintenance technician to decrease the time and effort required to detect and isolate malfunctions in digital circuit boards. This system incorporates sequencing and timing considerations of the test equipment in manual work-around prior to simulation. The simulator is not equipped to handle race, oscillations, and initialization problems automatically. The modeling technique utilizes MACRO's at the IC or component level. Internal component faults are not detected or isolated. Nine types of failures are detected by a stuck-at-1 (SA1) and stuck-at-0 (SA0) input and output to the board and components.

The input program performs editing or checking of model descriptions. The simulator outputs include: fault dictionary; programming worksheet; scope trace for output patterns; test statistics which indicate total external failures, external failures detected, percentage of detections, and dictionary graphic distributions, and average IC isolation resolution. Stimulus patterns for FLASH are generated manually.

This system is available on the MAR II Time Sharing System (GE-635 computer). Typical computer capacity requires 120K bytes of core memory on an IBM 370-145 computer for 100 IC boards and 512K bytes for 400 IC boards. This system is used by Micro Inc. to provide a commercial service to the electronics industry for the preparation of commercial digital board test programs. In addition, Sperry-Rand and General Radio are planning to use adaptations of this system.

TGEN

This digital test program generation system TGEN (Figure 9) was developed by RCA Princeton labs and is presently under evaluation in an RCA electronics facility in Van Nuys, California.

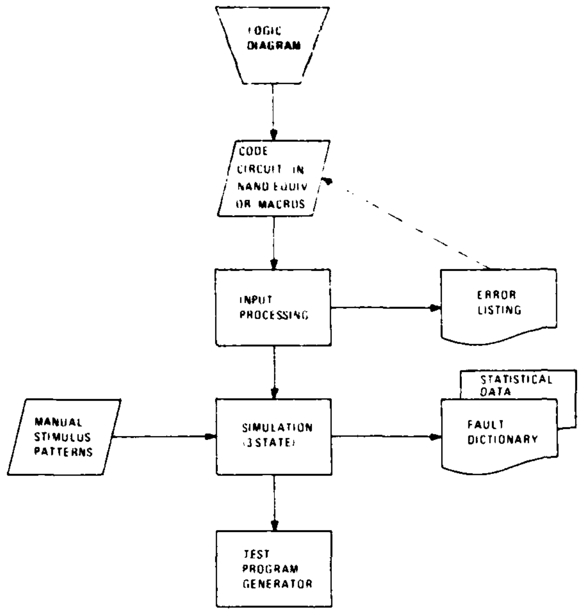

Figure 9 TGEN System Flow Diagram

The modeling technique utilizes both NAND equivalent modeling and MACRO subroutines to define digital circuit components. Stimulus patterns are prepared by test engineers manually and inputted to the simulator. The simulator utilizes three states, 1, 0, and X (unknown), and provides for race and hazard detection. Fault dictionaries are produced by the simulator based upon stuck-at-1 (SA1) and stuck-at-0 (SA0) inputs and outputs to the circuit board and its components. Internal and/or external component faults may be simulated. The simulator is capable of accepting digital circuits up to 500 logic elements in size. The TGEN system has been designed for and is operational on an RCA Spectra 70-61 computer.

FAULTS II

The Fault Analysis Using Logical Test Sequencing (FAULTS) system is a set of computer programs designed by General Dynamics to generate test programs for automatic test equipment to detect and isolate complex combinational and sequential digital circuits. The Faults II programs are written for the IBM 360 computer in Fortran IV (G) and IBM 360 assembler (F) and requires a minimum of 100K bytes of computer meoory and approximately one million bytes of disc storage.

The Faults system flow is presented in Figure 10. The digital circuit description is first coded in a concise input language utilizing MACRO's for circuit components.

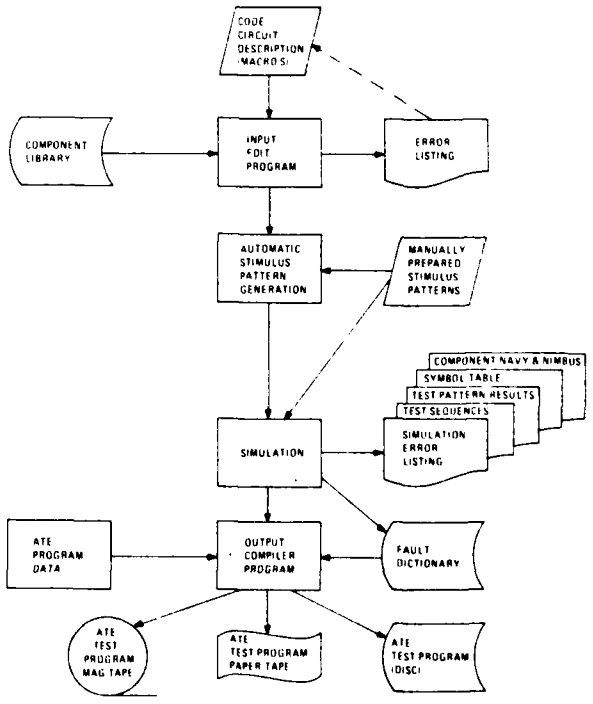

Figure 10 FAULTS System Flow Diagram

The coded circuit description is run through the input program to detect description errors. A component library is available to simplify circuit model development. The system has the capability of developing stimulus patterns utilizing three options:

a. Generate all stimulus patterns automatically

b. Accept manually prepared stimulus patterns

c. Accept both automatic and manually prepared stimulus patterns

The simulation portion of Faults II operates as follows. From the circuit description the system constructs two simulation models. The first is a true representation of the operational circuit and the second is a circuit model with the ability of inserting faults. The system then applies the input stimulus patterns until the outputs of the operational model and the faulted model differ. The input sequence that result in a difference between the two models is saved by the computer program and the others are discarded. The fault in the second circuit model is then changed and the input stimulus patterns are resumed. This simulation continues until all possible faults are detected, meaning all circuit components are completely exercised. If a defect cannot be found, a circuit redundancy is indicated. As a result of this circuit model faulting the input stimulus pattern sets and response pattern sets for each fault are produced and become part of the Fault dictionary. In addition, the system produces the following outputs: (1) listing of the input cards along with error messages generalized, (2) a listing of errors generated during simulation, (3) a listing of test sequences generated, (4) a listing of the results of each input stimulus pattern set, (5) a listing of the symbol table (cross-reference between node names and corresponding nodes numbers assigned), (6) a listing of component names and numbers. The system can identify circuit race and oscillation conditions, simulate circuit timing conditions, and provide for a limited initialization capability.

SALT

The Sequential Automatic Logic Test system was developed by IBM. The SALT system (Figure 11) is a three-state simulator (0, 1 and X) designed to generate test programs for combinational and sequential digital circuits. Man-machine

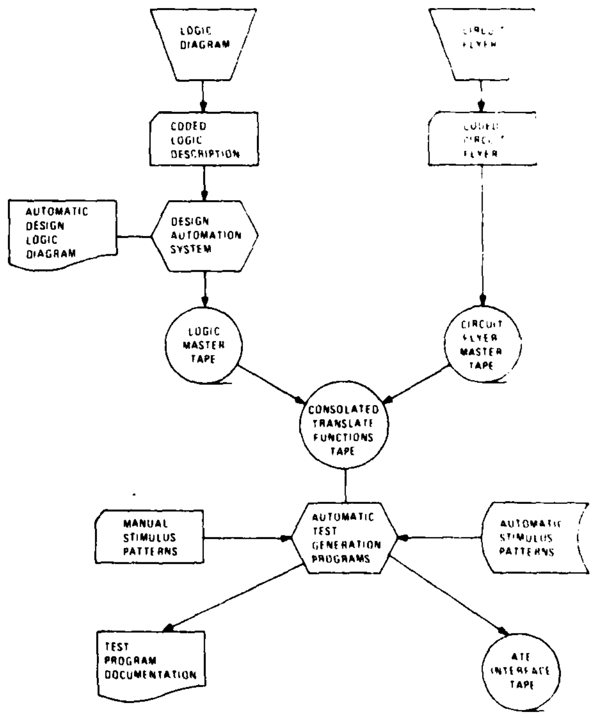
LOGIC DIAGRAM
CIRCUIT FLYER
CODED LOGIC DESCRIPTION
CODED CIRCUIT FLYER
AUTOMATIC DESIGN LOGIC DIAGRAM
DESIGN AUTOMATION SYSTEM
LOGIC MASTER TAPE
CIRCUIT FLYER MASTER TAPE
CONSOLATED TRANSLATE FUNCTIONS TAPE
MANUAL STIMULUS PATTERNS
AUTOMATIC TEST GENERATION PROGRAMS
AUTOMATIC STIMULUS PATTERNS
TEST PROGRAM DOCUMENTATION
ATE INTERFACE TAPE

Figure 11 SALT System Flow Diagram

nteractions are required to code ogic descriptions and to provide .nitialization of circuits. SALT loes provide for detecting race ınd oscillating conditions. The SALT simulation approach generates enough test patterns to detect all significant failures. Test patterns are generated based on single, repetitive failures for stuck-at-1 (SA1) and stuck-at-0 (SA0) failure modes. Each test pattern is simulated with all significant failures. Test pattern sets for each failure that differs from the normal or good response are saved and become a part of the test interface data. The ATE interface tape includes: (1) test patterns, (2) order of application, (3) good machine responses (GMR), (4) failing machine responses (FMR) and (5) FMR/failing component cross-reference (failure pointers). The system was designed for the IBM 360 computer. The system is not currently available for independent use.

TASC

The Terminal Access Simulation and Computation system was developed by Pacific Applied Systems Inc. The TASC system is an interactive system of computer programs written in FORTRAN IV. This system produces test input and response patterns for digital random logic circuits for test program fault detection and isolation. Based on a MACRO function simulation modeling technique, the TASC system simulates both sequential and combinatorial logic functions, including gates, flip-flops, latches and feedback loops, utilizing SSI, MSI and/or LSI integrated circuits.

The TASC system is composed of five basic modules, each independently acting, but all inter-related to the production of test programs and test procedures. The TASC system flow is presented in Figure 12.

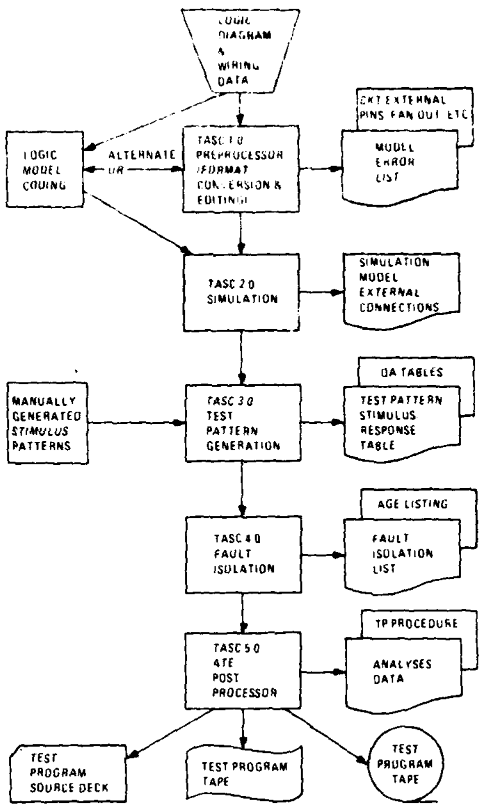
LOGIC DIAGRAM & WIRING DATA
LOGIC MODEL CODING
ALTERNATE OR
TASC 1.0 PREPROCESSOR (FORMAT CONVERSION & EDITING)
CKT EXTERNAL PINS FAN OUT ETC
MODEL ERROR LIST
TASC 2.0 SIMULATION
SIMULATION MODEL EXTERNAL CONNECTIONS
MANUALLY GENERATED STIMULUS PATTERNS
TASC 3.0 TEST PATTERN GENERATION
QA TABLES
TEST PATTERN STIMULUS RESPONSE TABLE
TASC 4.0 FAULT ISOLATION
AGE LISTING
FAULT ISOLATION LIST
TASC 5.0 ATE POST PROCESSOR
TP PROCEDURE
ANALYSES DATA
TEST PROGRAM SOURCE DECK
TEST PROGRAM TAPE
TEST PROGRAM TAPE

Figure 12 TASC System Flow Diagram

TASC 1.0 is a preprocessor used to reformat logic source data into the TASC format. If the data is available on computer-acceptable media, i.e., punched cards, paper tape or magnetic tape, TASC 1.0 buffers the data automatically, performs error detection, and provides a printout of circuit external pins, fan-out, logic types, logic count, etc. If the source data is not in an automated format, TASC 1.0 is replaced by manual encoding of the data into the required format.

TASC 2.0 is a simulation modeling program which, through processing of the functional and wiring data of TASC 1.0 constructs a MACRO function simulation model of the circuit. This model provides, as closely as possible, the relationship between the circuit nodes and the simulation model nodes.

TASC 3.0 is a stimulus pattern generator which produces an internodal status map of the condition (logical true or false) of each node for each input pattern, and a quality assurance table containing the condition of each node on a circuit function basis, which by inspection, assures a complete test. The stimulus pattern generator has the capability to automatically generate stimulus for most combinatorial elements. All sequential stimulus must be generated manually.

TASC 4.0 produces diagnostic fault isolation data based upon topological or path sensitization criteria.

TASC 5.0 is an output buffer which produces a completed diagnostic test procedure based upon the outputs of TASC 2.0, 3.0 and 4.0 in a format suitable to the user. The output may be in a form of a procedure, paper or magnetic tape, or a punched card deck in the users ATE source language.

ACKNOWLEDGEMENT

The author greatfully acknowledges the contribution of Harry D. Kimp in providing descriptions of the computer aided test design systems discussed in this paper. Much of the information was taken from the Naval Air Engineering Center Report GSED-MISC-107, S-3A Automatic Test Equipment and Automated Test Program Generation for Avionic Shop Replaceable Assemblies, dated February 1973 which was authored by Mr. Kimp as a consultant to the Naval Air Engineering Center, Philadelphia, PA

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By Per Ltr. on File	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DATE
ILMED
-8